Integrating Graph-Based and Transition-Based Dependency Parsers

Joakim Nivre
Växjö University
Computer Science
SE-35195 Växjö

Uppsala University
Linguistics and Philology
SE-75126 Uppsala
nivre@msi.vxu.se

Ryan McDonald
Google Inc.
76 Ninth Avenue
New York, NY 10011
ryanmcd@google.com

Abstract

Previous studies of data-driven dependency parsing have shown that the distribution of parsing errors are correlated with theoretical properties of the models used for learning and inference. In this paper, we show how these results can be exploited to improve parsing accuracy by integrating a graph-based and a transition-based model. By letting one model generate features for the other, we consistently improve accuracy for both models, resulting in a significant improvement of the state of the art when evaluated on data sets from the CoNLL-X shared task.

1 Introduction

Syntactic dependency graphs have recently gained a wide interest in the natural language processing community and have been used for many problems ranging from machine translation (Ding and Palmer, 2004) to ontology construction (Snow et al., 2005). A dependency graph for a sentence represents each word and its syntactic dependents through labeled directed arcs, as shown in figure 1. One advantage of this representation is that it extends naturally to discontinuous constructions, which arise due to long distance dependencies or in languages where syntactic structure is encoded in morphology rather than in word order. This is undoubtedly one of the reasons for the emergence of dependency parsers for a wide range of languages. Many of these parsers are based on data-driven parsing models, which learn to produce dependency graphs for sentences solely from an annotated corpus and can be easily ported to any language or domain in which annotated resources exist.

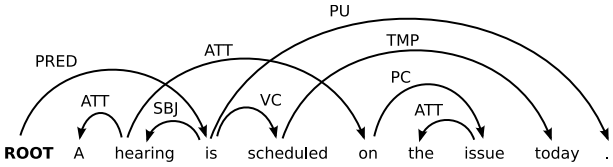

Figure 1: Dependency graph for an English sentence.

Practically all data-driven models that have been proposed for dependency parsing in recent years can be described as either *graph-based* or *transition-based* (McDonald and Nivre, 2007). In graph-based parsing, we learn a model for scoring possible dependency graphs for a given sentence, typically by factoring the graphs into their component arcs, and perform parsing by searching for the highest-scoring graph. This type of model has been used by, among others, Eisner (1996), McDonald et al. (2005a), and Nakagawa (2007). In transition-based parsing, we instead learn a model for scoring transitions from one parser state to the next, conditioned on the parse history, and perform parsing by greedily taking the highest-scoring transition out of every parser state until we have derived a complete dependency graph. This approach is represented, for example, by the models of Yamada and Matsumoto (2003), Nivre et al. (2004), and Attardi (2006).

Theoretically, these approaches are very different. The graph-based models are globally trained and use exact inference algorithms, but define features over a limited history of parsing decisions. The transition-based models are essentially the opposite. They use local training and greedy inference algorithms, but

Proceedings of ACL-08: HLT, pages 950–958,
Columbus, Ohio, USA, June 2008.

define features over a rich history of parsing decisions. This is a fundamental trade-off that is hard to overcome by tractable means. Both models have been used to achieve state-of-the-art accuracy for a wide range of languages, as shown in the CoNLL shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007), but McDonald and Nivre (2007) showed that a detailed error analysis reveals important differences in the distribution of errors associated with the two models.

In this paper, we consider a simple way of integrating graph-based and transition-based models in order to exploit their complementary strengths and thereby improve parsing accuracy beyond what is possible by either model in isolation. The method integrates the two models by allowing the output of one model to define features for the other. This method is simple – requiring only the definition of new features – and robust by allowing a model to learn relative to the predictions of the other.

2 Two Models for Dependency Parsing

2.1 Preliminaries

Given a set $L = \{l_1, \ldots, l_{|L|}\}$ of arc labels (dependency relations), a dependency graph for an input sentence $x = w_0, w_1, \ldots, w_n$ (where w_0 = ROOT) is a labeled directed graph $G = (V, A)$ consisting of a set of nodes $V = \{0, 1, \ldots, n\}$[1] and a set of labeled directed arcs $A \subseteq V \times V \times L$, i.e., if $(i, j, l) \in A$ for $i, j \in V$ and $l \in L$, then there is an arc from node i to node j with label l in the graph. A dependency graph G for a sentence x must be a directed tree originating out of the root node 0 and spanning all nodes in V, as exemplified by the graph in figure 1. This is a common constraint in many dependency parsing theories and their implementations.

2.2 Graph-Based Models

Graph-based dependency parsers parameterize a model over smaller substructures in order to search the space of valid dependency graphs and produce the most likely one. The simplest parameterization is the arc-factored model that defines a real-valued score function for arcs $s(i, j, l)$ and further defines the score of a dependency graph as the sum of the score of all the arcs it contains. As a result, the dependency parsing problem is written:

$$G = \arg\max_{G=(V,A)} \sum_{(i,j,l)\in A} s(i, j, l)$$

This problem is equivalent to finding the highest scoring directed spanning tree in the complete graph over the input sentence, which can be solved in $O(n^2)$ time (McDonald et al., 2005b). Additional parameterizations are possible that take more than one arc into account, but have varying effects on complexity (McDonald and Satta, 2007). An advantage of graph-based methods is that tractable inference enables the use of standard structured learning techniques that globally set parameters to maximize parsing performance on the training set (McDonald et al., 2005a). The primary disadvantage of these models is that scores – and as a result any feature representations – are restricted to a single arc or a small number of arcs in the graph.

The specific graph-based model studied in this work is that presented by McDonald et al. (2006), which factors scores over pairs of arcs (instead of just single arcs) and uses near exhaustive search for unlabeled parsing coupled with a separate classifier to label each arc. We call this system MSTParser, or simply MST for short, which is also the name of the freely available implementation.[2]

2.3 Transition-Based Models

Transition-based dependency parsing systems use a model parameterized over transitions of an abstract machine for deriving dependency graphs, such that every transition sequence from the designated initial configuration to some terminal configuration derives a valid dependency graph. Given a real-valued score function $s(c, t)$ (for transition t out of configuration c), parsing can be performed by starting from the initial configuration and taking the optimal transition $t^* = \arg\max_{t\in T} s(c, t)$ out of every configuration c until a terminal configuration is reached. This can be seen as a greedy search for the optimal dependency graph, based on a sequence of locally optimal decisions in terms of the transition system.

Many transition systems for data-driven dependency parsing are inspired by shift-reduce parsing,

[1] We use the common convention of representing words by their index in the sentence.

[2] http://mstparser.sourceforge.net

where each configuration c contains a stack σ_c for storing partially processed nodes and a buffer β_c containing the remaining input. Transitions in such a system add arcs to the dependency graph and manipulate the stack and buffer. One example is the transition system defined by Nivre (2003), which parses a sentence $x = w_0, w_1, \ldots, w_n$ in $O(n)$ time.

To learn a scoring function on transitions, these systems rely on discriminative learning methods, such as memory-based learning or support vector machines, using a strictly local learning procedure where only single transitions are scored (not complete transition sequences). The main advantage of these models is that features are not restricted to a limited number of graph arcs but can take into account the entire dependency graph built so far. The major disadvantage is that the greedy parsing strategy may lead to error propagation.

The specific transition-based model studied in this work is that presented by Nivre et al. (2006), which uses support vector machines to learn transition scores. We call this system MaltParser, or Malt for short, which is also the name of the freely available implementation.[3]

2.4 Comparison and Analysis

These models differ primarily with respect to three properties: *inference*, *learning*, and *feature representation*. MaltParser uses an inference algorithm that greedily chooses the best parsing decision based on the current parser history whereas MSTParser uses exhaustive search algorithms over the space of all valid dependency graphs to find the graph that maximizes the score. MaltParser trains a model to make a single classification decision (choose the next transition) whereas MSTParser trains a model to maximize the global score of correct graphs. MaltParser can introduce a rich feature history based on previous parser decisions, whereas MSTParser is forced to restrict features to a single decision or a pair of nearby decisions in order to retain efficiency.

These differences highlight an inherent trade-off between global inference/learning and expressiveness of feature representations. MSTParser favors the former at the expense of the latter and MaltParser the opposite. This difference was highlighted in the study of McDonald and Nivre (2007), which showed that the difference is reflected directly in the error distributions of the parsers. Thus, MaltParser is less accurate than MSTParser for long dependencies and those closer to the root of the graph, but more accurate for short dependencies and those farthest away from the root. Furthermore, MaltParser is more accurate for dependents that are nouns and pronouns, whereas MSTParser is more accurate for verbs, adjectives, adverbs, adpositions, and conjunctions.

Given that there is a strong negative correlation between dependency length and tree depth, and given that nouns and pronouns tend to be more deeply embedded than (at least) verbs and conjunctions, these patterns can all be explained by the same underlying factors. Simply put, MaltParser has an advantage in its richer feature representations, but this advantage is gradually diminished by the negative effect of error propagation due to the greedy inference strategy as sentences and dependencies get longer. MSTParser has a more even distribution of errors, which is expected given that the inference algorithm and feature representation should not prefer one type of arc over another. This naturally leads one to ask: Is it possible to integrate the two models in order to exploit their complementary strengths? This is the topic of the remainder of this paper.

3 Integrated Models

There are many conceivable ways of combining the two parsers, including more or less complex ensemble systems and voting schemes, which only perform the integration at parsing time. However, given that we are dealing with data-driven models, it should be possible to integrate at learning time, so that the two complementary models can learn from one another. In this paper, we propose to do this by letting one model generate features for the other.

3.1 Feature-Based Integration

As explained in section 2, both models essentially learn a scoring function $s : X \rightarrow \mathbb{R}$, where the domain X is different for the two models. For the graph-based model, X is the set of possible dependency arcs (i, j, l); for the transition-based model, X is the set of possible configuration-transition pairs (c, t). But in both cases, the input is represented

[3] http://w3.msi.vxu.se/∼jha/maltparser/

$\mathbf{MST_{Malt}}$ – defined over (i, j, l) ($*$ = any label/node)
Is $(i, j, *)$ in G_x^{Malt}?
Is (i, j, l) in G_x^{Malt}?
Is $(i, j, *)$ not in G_x^{Malt}?
Is (i, j, l) not in G_x^{Malt}?
Identity of l' such that $(*, j, l')$ is in G_x^{Malt}?
Identity of l' such that (i, j, l') is in G_x^{Malt}?
$\mathbf{Malt_{MST}}$ – defined over (c, t) ($*$ = any label/node)
Is $(\sigma_c^0, \beta_c^0, *)$ in G_x^{MST}?
Is $(\beta_c^0, \sigma_c^0, *)$ in G_x^{MST}?
Head direction for σ_c^0 in G_x^{MST} (left/right/ROOT)
Head direction for β_c^0 in G_x^{MST} (left/right/ROOT)
Identity of l such that $(*, \sigma_c^0, l)$ is in G_x^{MST}?
Identity of l such that $(*, \beta_c^0, l)$ is in G_x^{MST}?

Table 1: Guide features for MST_{Malt} and Malt_{MST}.

by a k-dimensional feature vector $\mathbf{f} : X \rightarrow \mathbb{R}^k$. In the feature-based integration we simply extend the feature vector for one model, called the *base model*, with a certain number of features generated by the other model, which we call the *guide model* in this context. The additional features will be referred to as *guide features*, and the version of the base model trained with the extended feature vector will be called the *guided model*. The idea is that the guided model should be able to learn in which situations to trust the guide features, in order to exploit the complementary strength of the guide model, so that performance can be improved with respect to the base parser. This method of combining classifiers is sometimes referred to as *classifier stacking*.

The exact form of the guide features depend on properties of the base model and will be discussed in sections 3.2–3.3 below, but the overall scheme for the feature-based integration can be described as follows. To train a guided version B_C of base model B with guide model C and training set T, the guided model is trained, not on the original training set T, but on a version of T that has been parsed with the guide model C under a cross-validation scheme (to avoid overlap with training data for C). This means that, for every sentence $x \in T$, B_C has access at training time to both the gold standard dependency graph G_x and the graph G_x^C predicted by C, and it is the latter that forms the basis for the additional guide features. When parsing a new sentence x' with B_C, x' is first parsed with model C (this time trained on the entire training set T) to derive $G_{x'}^C$, so that the guide features can be extracted also at parsing time.

3.2 The Guided Graph-Based Model

The graph-based model, MSTParser, learns a scoring function $s(i, j, l) \in \mathbb{R}$ over labeled dependencies. More precisely, dependency arcs (or pairs of arcs) are first represented by a high dimensional feature vector $\mathbf{f}(i, j, l) \in \mathbb{R}^k$, where $\mathbf{f}$ is typically a binary feature vector over properties of the arc as well as the surrounding input (McDonald et al., 2005a; McDonald et al., 2006). The score of an arc is defined as a linear classifier $s(i, j, l) = \mathbf{w} \cdot \mathbf{f}(i, j, l)$, where $\mathbf{w}$ is a vector of feature weights to be learned by the model.

For the guided graph-based model, which we call MST_{Malt}, this feature representation is modified to include an additional argument G_x^{Malt}, which is the dependency graph predicted by MaltParser on the input sentence x. Thus, the new feature representation will map an arc *and* the entire predicted MaltParser graph to a high dimensional feature representation, $\mathbf{f}(i, j, l, G_x^{\text{Malt}}) \in \mathbb{R}^{k+m}$. These m additional features account for the guide features over the MaltParser output. The specific features used by MST_{Malt} are given in table 1. All features are conjoined with the part-of-speech tags of the words involved in the dependency to allow the guided parser to learn weights relative to different surface syntactic environments. Though MSTParser is capable of defining features over pairs of arcs, we restrict the guide features over single arcs as this resulted in higher accuracies during preliminary experiments.

3.3 The Guided Transition-Based Model

The transition-based model, MaltParser, learns a scoring function $s(c, t) \in \mathbb{R}$ over configurations and transitions. The set of training instances for this learning problem is the set of pairs (c, t) such that t is the correct transition out of c in the transition sequence that derives the correct dependency graph G_x for some sentence x in the training set T. Each training instance (c, t) is represented by a feature vector $\mathbf{f}(c, t) \in \mathbb{R}^k$, where features are defined in terms of arbitrary properties of the configuration c, including the state of the stack σ_c, the input buffer β_c, and the partially built dependency graph G_c. In particular, many features involve properties of the two target tokens, the token on top of the stack σ_c (σ_c^0) and the first token in the input buffer β_c (β_c^0),

which are the two tokens that may become connected by a dependency arc through the transition out of c. The full set of features used by the base model MaltParser is described in Nivre et al. (2006).

For the guided transition-based model, which we call Malt_{MST}, training instances are extended to triples (c, t, G_x^{MST}), where G_x^{MST} is the dependency graph predicted by the graph-based MSTParser for the sentence x to which the configuration c belongs. We define m additional guide features, based on properties of G_x^{MST}, and extend the feature vector accordingly to $\mathbf{f}(c, t, G_x^{\text{MST}}) \in \mathbb{R}^{k+m}$. The specific features used by Malt_{MST} are given in table 1. Unlike MSTParser, features are not explicitly defined to conjoin guide features with part-of-speech features. These features are implicitly added through the polynomial kernel used to train the SVM.

4 Experiments

In this section, we present an experimental evaluation of the two guided models based on data from the CoNLL-X shared task, followed by a comparative error analysis including both the base models and the guided models. The data for the experiments are training and test sets for all thirteen languages from the CoNLL-X shared task on multilingual dependency parsing with training sets ranging in size from from 29,000 tokens (Slovene) to 1,249,000 tokens (Czech). The test sets are all standardized to about 5,000 tokens each. For more information on the data sets, see Buchholz and Marsi (2006).

The guided models were trained according to the scheme explained in section 3, with two-fold cross-validation when parsing the training data with the guide parsers. Preliminary experiments suggested that cross-validation with more folds had a negligible impact on the results. Models are evaluated by their *labeled attachment score* (LAS) on the test set, i.e., the percentage of tokens that are assigned both the correct head and the correct label, using the evaluation software from the CoNLL-X shared task with default settings.[4] Statistical significance was assessed using Dan Bikel's randomized parsing evaluation comparator with the default setting of 10,000 iterations.[5]

[4] http://nextens.uvt.nl/∼conll/software.html

[5] http://www.cis.upenn.edu/∼dbikel/software.html

Language	MST	MST_{Malt}	Malt	Malt_{MST}
Arabic	66.91	**68.64** (+1.73)	66.71	67.80 (+1.09)
Bulgarian	87.57	**89.05** (+1.48)	87.41	88.59 (+1.18)
Chinese	85.90	**88.43** (+2.53)	86.92	87.44 (+0.52)
Czech	80.18	**82.26** (+2.08)	78.42	81.18 (+2.76)
Danish	84.79	**86.67** (+1.88)	84.77	85.43 (+0.66)
Dutch	79.19	**81.63** (+2.44)	78.59	79.91 (+1.32)
German	87.34	**88.46** (+1.12)	85.82	87.66 (+1.84)
Japanese	90.71	91.43 (+0.72)	91.65	**92.20** (+0.55)
Portuguese	86.82	87.50 (+0.68)	87.60	**88.64** (+1.04)
Slovene	73.44	**75.94** (+2.50)	70.30	74.24 (+3.94)
Spanish	82.25	**83.99** (+1.74)	81.29	82.41 (+1.12)
Swedish	82.55	**84.66** (+2.11)	84.58	84.31 (–0.27)
Turkish	63.19	64.29 (+1.10)	65.58	**66.28** (+0.70)
Average	80.83	**82.53** (+1.70)	80.74	82.01 (+1.27)

Table 2: Labeled attachment scores for base parsers and guided parsers (improvement in percentage points).

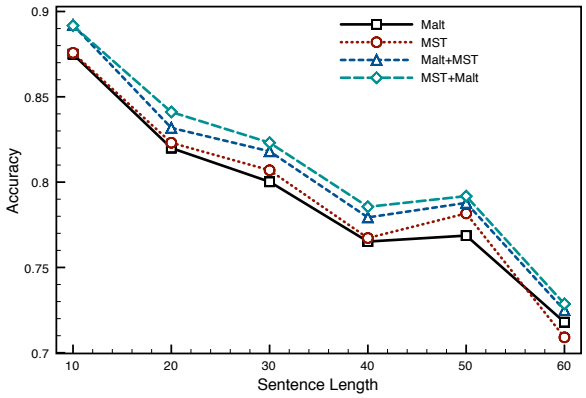

Figure 2: Accuracy relative to sentence length.

4.1 Results

Table 2 shows the results, for each language and on average, for the two base models (MST, Malt) and for the two guided models (MST_{Malt}, Malt_{MST}). First of all, we see that both guided models show a very consistent increase in accuracy compared to their base model, even though the extent of the improvement varies across languages from about half a percentage point (Malt_{MST} on Chinese) up to almost four percentage points (Malt_{MST} on Slovene).[6] It is thus quite clear that both models have the capacity to learn from features generated by the other model. However, it is also clear that the graph-based MST model shows a somewhat larger improvement, both on average and for all languages except Czech,

[6] The only exception to this pattern is the result for Malt_{MST} on Swedish, where we see an unexpected drop in accuracy compared to the base model.

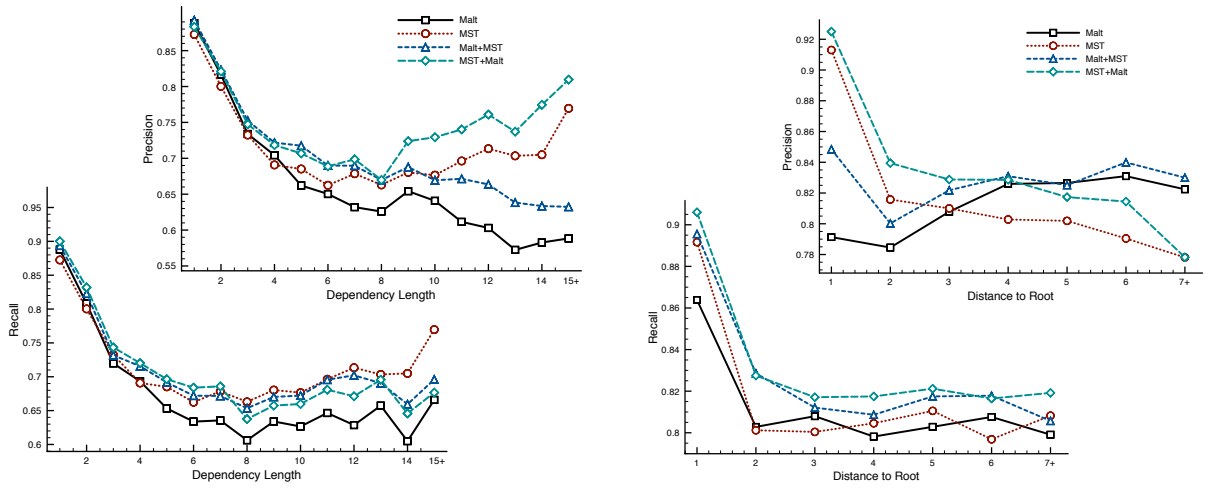

Figure 3: Dependency arc precision/recall relative to predicted/gold for (a) dependency length and (b) distance to root.

German, Portuguese and Slovene. Finally, given that the two base models had the previously best performance for these data sets, the guided models achieve a substantial improvement of the state of the art. While there is no statistically significant difference between the two base models, they are both outperformed by Malt_{MST} ($p < 0.0001$), which in turn has significantly lower accuracy than MST_{Malt} ($p < 0.0005$).

An extension to the models described so far would be to iteratively integrate the two parsers in the spirit of pipeline iteration (Hollingshead and Roark, 2007). For example, one could start with a Malt model, use it to train a guided MST_{Malt} model, then use that as the guide to train a $\text{Malt}_{\text{MST}_{\text{Malt}}}$ model, etc. We ran such experiments, but found that accuracy did not increase significantly and in some cases decreased slightly. This was true regardless of which parser began the iterative process. In retrospect, this result is not surprising. Since the initial integration effectively incorporates knowledge from both parsing systems, there is little to be gained by adding additional parsers in the chain.

4.2 Analysis

The experimental results presented so far show that feature-based integration is a viable approach for improving the accuracy of both graph-based and transition-based models for dependency parsing, but they say very little about how the integration benefits the two models and what aspects of the parsing process are improved as a result. In order to get a better understanding of these matters, we replicate parts of the error analysis presented by McDonald and Nivre (2007), where parsing errors are related to different structural properties of sentences and their dependency graphs. For each of the four models evaluated, we compute error statistics for labeled attachment over all twelve languages together.

Figure 2 shows accuracy in relation to sentence length, binned into ten-word intervals (1–10, 11-20, etc.). As expected, Malt and MST have very similar accuracy for short sentences but Malt degrades more rapidly with increasing sentence length because of error propagation (McDonald and Nivre, 2007). The guided models, Malt_{MST} and MST_{Malt}, behave in a very similar fashion with respect to each other but both outperform their base parser over the entire range of sentence lengths. However, except for the two extreme data points (0–10 and 51–60) there is also a slight tendency for Malt_{MST} to improve more for longer sentences and for MST_{Malt} to improve more for short sentences, which indicates that the feature-based integration allows one parser to exploit the strength of the other.

Figure 3(a) plots precision (top) and recall (bottom) for dependency arcs of different lengths (predicted arcs for precision, gold standard arcs for recall). With respect to recall, the guided models appear to have a slight advantage over the base mod-

Part of Speech	MST	MST_{Malt}	Malt	Malt_{MST}
Verb	82.6	**85.1** (2.5)	81.9	84.3 (2.4)
Noun	80.0	81.7 (1.7)	80.7	**81.9** (1.2)
Pronoun	88.4	**89.4** (1.0)	89.2	89.3 (0.1)
Adjective	89.1	**89.6** (0.5)	87.9	89.0 (1.1)
Adverb	78.3	**79.6** (1.3)	77.4	78.1 (0.7)
Adposition	69.9	**71.5** (1.6)	68.8	70.7 (1.9)
Conjunction	73.1	**74.9** (1.8)	69.8	72.5 (2.7)

Table 3: Accuracy relative to dependent part of speech (improvement in percentage points).

els for short and medium distance arcs. With respect to precision, however, there are two clear patterns. First, the graph-based models have better precision than the transition-based models when predicting long arcs, which is compatible with the results of McDonald and Nivre (2007). Secondly, both the guided models have better precision than their base model and, for the most part, also their guide model. In particular MST_{Malt} outperforms MST and is comparable to Malt for short arcs. More interestingly, Malt_{MST} outperforms both Malt and MST for arcs up to length 9, which provides evidence that Malt_{MST} has learned specifically to trust the guide features from MST for longer dependencies. The reason that accuracy does not improve for dependencies of length greater than 9 is probably that these dependencies are too rare for Malt_{MST} to learn from the guide parser in these situations.

Figure 3(b) shows precision (top) and recall (bottom) for dependency arcs at different distances from the root (predicted arcs for precision, gold standard arcs for recall). Again, we find the clearest patterns in the graphs for precision, where Malt has very low precision near the root but improves with increasing depth, while MST shows the opposite trend (McDonald and Nivre, 2007). Considering the guided models, it is clear that Malt_{MST} improves in the direction of its guide model, with a 5-point increase in precision for dependents of the root and smaller improvements for longer distances. Similarly, MST_{Malt} improves precision in the range where its base parser is inferior to Malt and for distances up to 4 has an accuracy comparable to or higher than its guide parser Malt. This again provides evidence that the guided parsers are learning from their guide models.

Table 3 gives the accuracy for arcs relative to dependent part-of-speech. As expected, we see that MST does better than Malt for all categories except nouns and pronouns (McDonald and Nivre, 2007). But we also see that the guided models in all cases improve over their base parser and, in most cases, also over their guide parser. The general trend is that MST improves more than Malt, except for adjectives and conjunctions, where Malt has a greater disadvantage from the start and therefore benefits more from the guide features.

Considering the results for parts of speech, as well as those for dependency length and root distance, it is interesting to note that the guided models often improve even in situations where their base parsers are more accurate than their guide models. This suggests that the improvement is not a simple function of the raw accuracy of the guide model but depends on the fact that labeled dependency decisions interact in inference algorithms for both graph-based and transition-based parsing systems. Thus, if a parser can improve its accuracy on one class of dependencies, e.g., longer ones, then we can expect to see improvements on all types of dependencies – as we do.

The interaction between different decisions may also be part of the explanation why MST benefits more from the feature-based integration than Malt, with significantly higher accuracy for MST_{Malt} than for Malt_{MST} as a result. Since inference is global (or practically global) in the graph-based model, an improvement in one type of dependency has a good chance of influencing the accuracy of other dependencies, whereas in the transition-based model, where inference is greedy, some of these additional benefits will be lost because of error propagation. This is reflected in the error analysis in the following recurrent pattern: Where Malt does well, Malt_{MST} does only slightly better. But where MST is good, MST_{Malt} is often significantly better.

Another part of the explanation may have to do with the learning algorithms used by the systems. Although both Malt and MST use discriminative algorithms, Malt uses a batch learning algorithm (SVM) and MST uses an online learning algorithm (MIRA). If the original rich feature representation of Malt is sufficient to separate the training data, regularization may force the weights of the guided features to be small (since they are not needed at training time). On the other hand, an online learn-

ing algorithm will recognize the guided features as strong indicators early in training and give them a high weight as a result. Features with high weight early in training tend to have the most impact on the final classifier due to both weight regularization and averaging. This is in fact observed when inspecting the weights of MST_{Malt}.

5 Related Work

Combinations of graph-based and transition-based models for data-driven dependency parsing have previously been explored by Sagae and Lavie (2006), who report improvements of up to 1.7 percentage points over the best single parser when combining three transition-based models and one graph-based model for unlabeled dependency parsing, evaluated on data from the Penn Treebank. The combined parsing model is essentially an instance of the graph-based model, where arc scores are derived from the output of the different component parsers. Unlike the models presented here, integration takes place only at parsing time, not at learning time, and requires at least three different base parsers. The same technique was used by Hall et al. (2007) to combine six transition-based parsers in the best performing system in the CoNLL 2007 shared task.

Feature-based integration in the sense of letting a subset of the features for one model be derived from the output of a different model has been exploited for dependency parsing by McDonald (2006), who trained an instance of MSTParser using features generated by the parsers of Collins (1999) and Charniak (2000), which improved unlabeled accuracy by 1.7 percentage points, again on data from the Penn Treebank. In addition, feature-based integration has been used by Taskar et al. (2005), who trained a discriminative word alignment model using features derived from the IBM models, and by Florian et al. (2004), who trained classifiers on auxiliary data to guide named entity classifiers.

Feature-based integration also has points in common with co-training, which have been applied to syntactic parsing by Sarkar (2001) and Steedman et al. (2003), among others. The difference, of course, is that standard co-training is a weakly supervised method, where guide features *replace*, rather than *complement*, the gold standard annotation during training. Feature-based integration is also similar to parse re-ranking (Collins, 2000), where one parser produces a set of candidate parses and a second-stage classifier chooses the most likely one. However, feature-based integration is not explicitly constrained to any parse decisions that the guide model might make and only the single most likely parse is used from the guide model, making it significantly more efficient than re-ranking.

Finally, there are several recent developments in data-driven dependency parsing, which can be seen as targeting the specific weaknesses of graph-based and transition-based models, respectively, though without integrating the two models. Thus, Nakagawa (2007) and Hall (2007) both try to overcome the limited feature scope of graph-based models by adding global features, in the former case using Gibbs sampling to deal with the intractable inference problem, in the latter case using a re-ranking scheme. For transition-based models, the trend is to alleviate error propagation by abandoning greedy, deterministic inference in favor of beam search with globally normalized models for scoring transition sequences, either generative (Titov and Henderson, 2007a; Titov and Henderson, 2007b) or conditional (Duan et al., 2007; Johansson and Nugues, 2007).

6 Conclusion

In this paper, we have demonstrated how the two dominant approaches to data-driven dependency parsing, graph-based models and transition-based models, can be integrated by letting one model learn from features generated by the other. Our experimental results show that both models consistently improve their accuracy when given access to features generated by the other model, which leads to a significant advancement of the state of the art in data-driven dependency parsing. Moreover, a comparative error analysis reveals that the improvements are largely predictable from theoretical properties of the two models, in particular the tradeoff between global learning and inference, on the one hand, and rich feature representations, on the other. Directions for future research include a more detailed analysis of the effect of feature-based integration, as well as the exploration of other strategies for integrating different parsing models.

References

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of CoNLL*, pages 166–170.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*, pages 132–139.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML*, pages 175–182.

Yuan Ding and Martha Palmer. 2004. Synchronous dependency insertion grammars: A grammar formalism for syntax based statistical MT. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar*, pages 90–97.

Xiangyu Duan, Jun Zhao, and Bo Xu. 2007. Probabilistic parsing action models for multi-lingual dependency parsing. In *Proceedings of EMNLP-CoNLL*, pages 940–946.

Jason M. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345.

Radu Florian, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of NAACL/HLT*.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of EMNLP-CoNLL*.

Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of ACL*, pages 392–399.

Kristy Hollingshead and Brian Roark. 2007. Pipeline iteration. In *Proceedings of ACL*, pages 952–959.

Richard Johansson and Pierre Nugues. 2007. Incremental dependency parsing using online learning. In *Proceedings of EMNLP-CoNLL*, pages 1134–1138.

Ryan McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, pages 122–131.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, pages 122–131.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, pages 216–220.

Ryan McDonald. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Tetsuji Nakagawa. 2007. Multilingual dependency parsing using global features. In *Proceedings of EMNLP-CoNLL*, pages 952–956.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, pages 221–225.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of EMNLP-CoNLL*, pages 915–932.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of IWPT*, pages 149–160.

Kenji Sagae and Alon Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL: Short Papers*, pages 129–132.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*, pages 175–182.

Rion Snow, Dan Jurafsky, and Andrew Y. Ng. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of NIPS*.

Mark Steedman, Rebecca Hwa, Miles Osborne, and Anoop Sarkar. 2003. Corrected co-training for statistical parsers. In *Proceedings of ICML*, pages 95–102.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT/EMNLP*, pages 73–80.

Ivan Titov and James Henderson. 2007a. Fast and robust multilingual dependency parsing with a generative latent variable model. In *Proceedings of EMNLP-CoNLL*, pages 947–951.

Ivan Titov and James Henderson. 2007b. A latent variable model for generative dependency parsing. In *Proceedings of IWPT*, pages 144–155.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206.